US-PAT-NO: 5974503
DOCUMENT-IDENTIFIER: US 5974503 A
TITLE: Storage and access of continuous media files indexed as lists of raid stripe sets associated with file names

---

Application Filing Date - AD (1):

19970505

Brief Summary Text - BSTX (11):

The computational inefficiencies, data availability problems, and load balancing problems associated with storing continuous media in a conventional RAID storage system result from the relatively large size, relatively high access rate, and non-uniform access frequency of continuous media files. Because continuous media files are typically accessed in a sequential or isochronous fashion, it is desirable to access a relatively large transfer unit of continuous media data during each disk access due to the fixed overhead of managing each disk access. However, it is very likely that one continuous media file, such as a file for a popular movie, could exceed the capacity of a single disk drive, and could have an access frequency exceeding the combined access frequency of all other files stored in the RAID set. Moreover, in video-on-demand applications, multiple users or clients could be simultaneously accessing different portions of the same continuous media file. Therefore, for high data availability, it is desirable to distribute the continuous media data for each file over all of the disk drives in each RAID set, and for load balancing, it is also desirable to distribute the parity information for each continuous media file over a multiplicity of the disk drives.

Brief Summary Text - BSTX (17):

In the preferred Embodiment, each stripe set is assigned an identification number from which can be computed in sequence the starting block addresses of the transfer units in the stripe set. The continuous media data is logically organized as individually named files called "clips". A clip directory associates a clip identification number with a list of allocated stripe sets.

Brief Summary Text - BSTX (18):

The stripe set list associated with each clip, for example, includes a doubly-linked list of entries, and each entry includes a starting stripe set number, an ending stripe set number, and a value indicating the number of data blocks included in the terminal stripe set. Therefore, each entry in the list represents in sequence data blocks beginning in the initial stripe set, continuing in any intermediate stripe set, and ending in the terminal stripe set, and including in the terminal stripe set only the indicated number of data blocks. The stripe set list for each clip can therefore easily be edited by linking and unlinking entries. When editing of the clip results in a number of stripe sets that are partially empty, compaction can be performed as a background operation by copying data to newly allocated stripe sets, unlinking the entries pointing to the old stripe sets, linking to new entries pointing to the newly allocated stripe sets, and deallocating the old stripe sets. Stripe sets are allocated by removing them from a stripe set free list, and de-allocated by returning them to the stripe set free list. Each entry in the stripe set free list, for example, includes a starting

stripe set number and an ending stripe set number, to indicate a range of unallocated stripe sets.

Detailed Description Text - DETX (32):

Each disk in the CMFS volume set is divided into two areas: the data area and the inode area. The data area is used to store file data, while the inode area is used to store inodes that hold file metadata. In addition to the standard file metadata information, the inode contains an array of extent descriptors that locate each extent comprising the corresponding file. An extent descriptor may also point to an inode located on another disk. Such a descriptor is used to point to a continuation inode when a CMFS file spans multiple disks.

Detailed Description Text - DETX (124):

Turning now to FIG. 20, there is shown a block diagram illustrating the distribution of software used in the file server (20 in FIG. 1) for the "on-line" tape backup operations of FIG. 19. The backup software used for this purpose can be designed and written specifically for it, or it can be a modification of an existing backup package, as described below. In particular, an existing implementation of suitable backup software is adapted from the Epoch (trademark) backup software sold by EMC Corporation, 171 South Street, Hopkinton, Mass. 01748. The backup software includes a backup scheduler 201, a volume manager 202, and a save/restore data mover 203. The backup software in the file server (20 in FIG. 1) is adapted from the Epoch (trademark) Hierarchical Storage Management (HSM) software by splitting the save/restore data mover 203 from the backup scheduler 201 and volume manager 202 so that the data mover 203 can run in the environment of a separate computer. The backup scheduler 201 and the volume manager 202 comprise the "control" part of the Epoch (trademark) backup software. The backup scheduler 201 and the volume manager 202 run in the active controller server (28 or 29 in FIG. 2) to provide backup scheduling, migration and catalog management. Alternatively, the backup scheduler 201 and the volume manager 202 could run in a separate external computer (not shown), which could communicate with the stream servers 21 over a network different from the internal Ethernet 26. The save/restore data mover 203 is replicated in each of the stream servers 21, only one of which is shown in FIG. 20. The save/restore data mover 203 is responsive to commands transmitted by the backup scheduler 201 and volume manager 202 over the internal Ethernet link 26. The backup scheduler 201, the volume manager 202, and the save/restore data mover 203 are adapted to communicate via the commands over the Ethernet link 26 instead of the procedure calls that are used in the Epoch (trademark) backup software.

Detailed Description Text - DETX (135):

As shown in FIG. 21, data for one continuous media file, shown with cross-hatching, are striped across all of the disk drives in the RAID set and have associated parity in each of the disk drives that store parity in the RAID set. Such striping across all of the disk drives in the RAID set, in combination with a relatively fixed overhead for disk access and the sequential or isochronous fashion of data access to the multimedia file, however, leads to a problem of a high rate of disk access unless a relatively large transfer unit of continuous media data is accessed during each disk access.

Detailed Description Text - DETX (152):

The stripe set list associated with each clip, for example, includes a doubly-linked list of entries, and each entry includes a starting stripe set number, an ending stripe set number, and a value indicating the number of data blocks included in the terminal stripe set. Therefore, each entry in the list represents in

sequence data blocks beginning in the initial stripe set, continuing in any intermediate stripe set, and ending in the terminal stripe set, and including in the terminal stripe set only the indicated number of data blocks. The stripe set list for each clip can therefore easily be edited by linking and unlinking entries.

Detailed Description Text - DETX (156):

As shown in FIG. 27, the video file server maintains an active client list 301 in order to manage the servicing of client requests. The active client list 301 is essentially a directory to blocks of information about maintaining respective isochronous data streams to the active clients. Such a block of information 302 includes a client identifier 303 identifying the client to which the block of information is relevant, a stream server identifier 304 identifying a stream server assigned to service the client, a play list 305 of clips that are transmitted in sequence to the client, and a current play position 306 in the play list and in the clip currently being played. The play list 305, for example, is a doubly-linked list including, at the head of the list, the clip identifier of the clip currently being transmitted to the client. The video file server responds to a client request for scheduling additional clips by inserting corresponding clip identifiers to the tail of the play list. The video file server responds to a client request to edit its schedule of clips by linking or unlinking corresponding clip identifiers to or from the client's play list.

Detailed Description Text - DETX (211):

Upon receipt of this command, the video file server returns information about clips and/or directories at the current position within the name space which are playable. Since there may be more entries than will fit within a single RPC response message, the information is returned over a sequence of calls. The cookie value is initialized to zero for the first call which returns some number of entries together with a new cookie value which should be sent in the next call. This sequence continues until a request returns with the "endoflist" field TRUE indicating that this request completes the information return.

Detailed Description Text - DETX (272):

In view of the above, there has been described a way of striping a sequence of continuous media data across parity groups in a RAID set in order to provide high data availability and load balancing. The striping distributes the sequence of continuous media data across all of the disk drives in the RAID set. The sequence of continuous media data is distributed across parity groups including parity in the disk drives in the RAID set that store parity, and in each of the parity groups, across the disk drives containing data. Because the sequence of continuous media data is comprised of contiguous transfer units in each parity group, at most one write access to the parity in each parity group need be performed during write access to each parity group.

Claims Text - CLTX (1):

1. A method of striping a sequence of continuous media data across a plurality of n disk drives in a RAID set storing data and associated parity across the disk drives, wherein the data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, parity is distributed among the disk drives in the RAID set and is stored in at least one hyper-volume of each of m disk drives in the RAID set, and transfer units of data storage in the n disk drives are associated with the sequence of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in

the disk drives and the columns of the matrix represent the data storage in the respective n disk drives in the RAID set, which further includes allocating data storage in the RAID set for the stream of continuous media data in fixed size stripe sets, each stripe set comprising a segment of the sequence of continuous media data distributed across each of the disk drives and each of the parity groups of hyper-volumes in the disk drives,

Claims Text - CLTX (3):

2. A method of striping a sequence of continuous media data across a plurality of n disk drives in a RAID set storing data and associated parity across the disk drives, wherein the data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, parity is distributed among the disk drives in the RAID set and is stored in at least one hyper-volume of each of m disk drives in the RAID set, and transfer units of data storage in the n disk drives are associated with the sequence of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent the data storage in the respective n disk drives in the RAID set, which further includes allocating data storage in the RAID set for the stream of continuous media data in fixed size stripe sets, each stripe set comprising a segment of the sequence of continuous media data distributed across each of the disk drives and each of the parity groups of hyper-volumes in the disk drives, and which includes associating a name of a continuous media data file with a list of the stripe sets.

Claims Text - CLTX (4):

3. The method as claimed in claim 1, which includes computing parity for each of the parity groups by computing parity changes for different ones of the disk drives due to storing of the sequence of continuous media data in the disk drives, reading the parity once from each of the parity groups, applying the parity changes for the different ones of the disk drives to the parity read once for each of the parity groups to produce modified parity, and writing the modified parity back to said each of the parity groups.

Claims Text - CLTX (7):

b) accessing the list of stripe sets to determine a sequence of the stripe sets storing the continuous media data; and

Claims Text - CLTX (8):

c) accessing the RAID set of disk drives to retrieve the sequence of stripe sets.

Claims Text - CLTX (13):

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in the disk drives;

Claims Text - CLTX (14):

wherein the sequence of continuous media data and associated parity is striped across the disk drives in the RAID set, the data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, parity is distributed among the disk drives in the RAID set and is contained in at least one

hyper-volume of each of the m disk drives in the RAID set, and transfer units of data storage in the n disk drives are associated with the sequence of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent the data storage in the respective n disk drives in the RAID set,

Claims Text - CLTX (15):

wherein data storage in the RAID set for the stream of continuous media data is allocated in fixed size stripe sets, each stripe set comprising a segment of the sequence of continuous media data distributed across each of the disk drives and each of the parity groups of hyper-volumes in the disk drives, and wherein the transfer unit has a predetermined size, and each stripe set includes (m)(n-1) transfer units of data.

Claims Text - CLTX (18):

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in the disk drives;

Claims Text - CLTX (19):

wherein the sequence of continuous media data and associated parity is striped across the disk drives in the RAID set, the data storage of each disk drive in the RAID set is partitioned into an integer number m of hyper-volumes, parity is distributed among the disk drives in the RAID set and is contained in at least one hyper-volume of each of the m disk drives in the RAID set, and transfer units of data storage in the n disk drives are associated with the sequence of continuous media data in a right-to-left and then top-to-bottom order in which the transfer units appear in an m row by n column matrix in which the rows of the matrix represent parity groups of hyper-volumes in the disk drives and the columns of the matrix represent the data storage in the respective n disk drives in the RAID set, wherein data storage in the RAID set for the stream of continuous media data is allocated in fixed size stripe sets, each stripe set comprising a segment of the sequence of continuous media data distributed across each of the disk drives and each of the parity groups of hyper-volumes in the disk drives, and wherein the controller has a directory associating a name of a continuous media data file with a list of the stripe sets.

Claims Text - CLTX (20):

9. The data storage system as claimed in claim 7, wherein the controller is programmed to compute parity for each of the parity groups by computing parity changes for different ones of the disk drives due to storing of the sequence of continuous media data in the disk drives, to read the parity once from each of the parity groups, to apply the parity changes for the different ones of the disk drives to the parity read once for each of the parity groups to produce modified parity, and to write the modified parity back to said each of the parity groups.

Claims Text - CLTX (23):

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in stripe sets in a RAID set of disk drives, the controller having a directory of continuous media data files and lists of the stripe sets associated with each of the continuous media data files, the controller being programmed to respond to a request to access a specified continuous media data file

by accessing the directory to find a list of stripe sets associated with the specified continuous media data file, to access the list of stripe sets to determine a sequence of the stripe sets storing the continuous media data, and to access the RAID set of disk drives to retrieve the sequence of stripe sets.

Claims Text - CLTX (26):

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in stripe sets in a RAID set of disk drives, the controller having a directory of continuous media data files and lists of the stripe sets associated with each of the continuous media data files, the controller being programmed to respond to a request to access a specified continuous media data file by accessing the directory to find a list of stripe sets associated with the specified continuous media data file, to access the list of stripe sets to determine a sequence of the stripe sets storing the continuous media data, and to access the RAID set of disk drives to retrieve the sequence of stripe sets, wherein the list of stripe sets includes at least one entry specifying a starting stripe set number and an ending stripe set number, and the controller is programmed to access the RAID set of disk drives to retrieve a stripe set identified by the starting stripe set number, to retrieve stripe sets identified by stripe set numbers between the starting stripe set number and the ending stripe set number, and to retrieve a stripe set identified by the ending stripe set number.

Claims Text - CLTX (29):

a controller coupled to the disk drives for accessing a sequence of continuous media data stored in stripe sets in a RAID set of disk drives, the controller having a directory of continuous media data files and lists of the stripe sets associated with each of the continuous media data files, the controller being programmed to respond to a request to access a specified continuous media data file by accessing the directory to find a list of stripe sets associated with the specified continuous media data file, to access the list of stripe sets to determine a sequence of the stripe sets storing the continuous media data, and to access the RAID set of disk drives to retrieve the sequence of stripe sets, and wherein the stripe set is comprised of a series of transfer units in disk drives of the RAID set, and the controller is programmed to sequentially advance an index and access a look-up table with the index in order to determine a disk drive in the RAID set containing a next transfer unit to be accessed.

US-PAT-NO: 5737747
DOCUMENT-IDENTIFIER: US 5737747 A
TITLE: Prefetching to service multiple video streams from an integrated cached disk array

---

Application Filing Date - AD (1):

19960610

Brief Summary Text - BSTX (20):

In a preferred embodiment, the video file server further includes a scheduler responsive to a client request for a video data stream from a data set such as a data set encoding a movie. The scheduler schedules the issuance of prefetch commands to the cached disk storage substation a certain time in advance of the corresponding fetch commands. The scheduler determines whether or not a video stream is already being supplied by the cached disk storage substation from the data set. If a video stream is already being supplied by the cached disk storage substation from the data set, then the scheduler decides whether to schedule the prefetch commands a minimum time in advance of the fetch commands in order to conserve cache resources, or whether to schedule the prefetch commands some greater time in advance of the fetch commands in order to conserve disk resources. In any case, if sufficient resources are not available to service the client request, then the request is denied.

Detailed Description Text - DETX (32):

Each disk in the CMFS volume set is divided into two areas: the data area and the inode area. The data area is used to store file data, while the inode area is used to store inodes that hold file metadata. In addition to the standard file metadata information, the inode contains an array of extent descriptors that locate each extent comprising the corresponding file. An extent descriptor may also point to an inode located on another disk. Such a descriptor is used to point to a continuation inode when a CMFS file spans multiple disks.

Detailed Description Text - DETX (124):

Turning now to FIG. 20, there is shown a block diagram illustrating the distribution of software used in the file server (20 in FIG. 1) for the "on-line" tape backup operations of FIG. 19. The backup software used for this purpose can be designed and written specifically for it, or it can be a modification of an existing backup package, as described below. In particular, an existing implementation of suitable backup software is adapted from the Epoch (trademark) backup software sold by EMC Corporation, 171 South Street, Hopkinton, Mass. 01748. The backup software includes a backup scheduler 201, a volume manager 202, and a save/restore data mover 203. The backup software in the file server (20 in FIG. 1) is adapted from the Epoch (trademark) Hierarchical Storage Management (HSM) software by splitting the save/restore data mover 203 from the backup scheduler 201 and volume manager 202 so that the data mover 203 can run in the environment of a separate computer. The backup scheduler 201 and the volume manager 202 comprise the "control" part of the Epoch (trademark) backup software. The backup scheduler 201 and the volume manager 202 run in the active controller server (28 or 29 in FIG. 2) to provide backup scheduling, migration and catalog management. Alternatively, the

backup scheduler 201 and the volume manager 202 could run in a separate external computer (not shown); which could communicate with the stream servers 21 over a network different from the internal Ethernet 26. The save/restore data mover 203 is replicated in each of the stream servers 21, only one of which is shown in FIG. 20. The save/restore data mover 203 is responsive to commands transmitted by the backup scheduler 201 and volume manager 202 over the internal Ethernet link 26. The backup scheduler 201, the volume manager 202, and the save/restore data mover 203 are adapted to communicate via the commands over the Ethernet link 26 instead of the procedure calls that are used in the Epoch (trademark) backup software.

Claims Text - CLTX (18):

10. The method as claimed in claim 9, wherein the scheduler checks whether the video file server is already supplying a video data stream from the data set, and when the video file server is already supplying a video data stream from the data set, the scheduler decides either to schedule said each prefetch command for said each process a minimum time in advance of said each fetch command for said each process in order to conserve cache resources, or to schedule said each prefetch command for said each process some greater time in advance of said each fetch command for said each process in order to conserve disk resources.

Claims Text - CLTX (33):

19. A cached disk storage subsystem responsive to a respective sequence of a prefetch command and a fetch command issued by each of a plurality of processes, said cached disk storage subsystem comprising:

Claims Text - CLTX (43):

a controller server responsive to requests from network clients for video data sets for initiating execution of a process for satisfying each request by issuing a sequence of prefetch and fetch commands for prefetching and fetching specified data segments of each video data set; and

Claims Text - CLTX (54):

a controller server responsive to requests from network clients for video data sets and initiating execution of a process for satisfying each request, each process issuing a sequence of prefetch and fetch commands for prefetching and fetching specified data segments of each video data set; and

US-PAT-NO: 6330572
DOCUMENT-IDENTIFIER: US 6330572 B1
TITLE: Hierarchical data storage management

Application Filing Date - AD (1):

19990715

Brief Summary Text - BSTX (13):

Also, a media independence feature can be incorporated whereby data can be stored on particular volumes without knowledge of the media type of the volume. In this case, the system and method provide an application programming interface that is substantially identical for all media types, for both direct access and sequential access, and for removable and non-removable media.

Brief Summary Text - BSTX (15):

The self-describing media feature can be implemented by storing volume metadata on each physical volume and by storing file metadata for each data file on the same physical volume on which the data file is stored. Thus, each file may include two files: a blob file with the actual file contents and a metadata file with identifying information. The metadata provides sufficient information to rebuild an inventory of the contents of a volume without access to the original file location database.

Brief Summary Text - BSTX (16):

The metadata can be useful in tracking volumes and files, verification of the identities of loaded volumes, and database recovery. The metadata can be specified by a client application and can be substantially transparent to the storage management server, providing information known to the client application such as the particular client applications and users associated with the volume or file.

Brief Summary Text - BSTX (17):

For unique identification of volumes and files, the metadata may incorporate a global unique identifier (guid) that is unique on a single server, as well as across a multi-server system. Also, a guid can be generated for each storage device. The guid can be used to track the location of a volume or file in the event the volume is moved across the system, or the file is moved across the various media volumes on a server or across the system. The guid for a particular file can be generated by the storage management system, but preferably is generated by the client application along with other metadata useful in referencing the file.

Brief Summary Text - BSTX (30):

Thus, the client application supporting the web-based workflow converts each acquired image to the supported format such that the user can have direct access to the images maintained by the storage management function within the web browser. No images need to be kept directly within the client database. The screen resolution and thumbnail versions can be "locked" down on the short-term media, such as a RAID, and never allowed to migrate offline to tape. At the same time, the high

resolution image may be allowed to migrate according to user-defined migration policies then in effect. In this manner, internet access to the images is as quick as possible. Also, the locked-down images are not stored by the database server, but rather by the HSM server for direct access by the user. By migrating the high resolution image, however, storage costs for RAID are dramatically reduced. In other words, an application can be configured to store images as economically and efficiently as possible using the system and method of the present invention, with the potential for growth of storage capacity being unlimited and scalable.

Brief Summary Text - BSTX (32):

This vault can be password and login protected. All image transmissions can be done under SSL. The vault image viewing is also secured through a virtual directory service, another sequence of logins into the storage management system, a Microsoft SQL Server, and the Windows NT NTFS file system itself. From the vault, the consumer can proof and create his/her own media order. That media order is placed into a "shopping basket" for the consumer, and totaled for payment and shipping. The consumer may also send those images to a third party via internet mail. Those images stored within the vault are set on an aging algorithm where after a predetermined number of days, the images are deleted from the system.

Brief Summary Text - BSTX (35):

In an added embodiment, the present invention provides a method for storing files, the method comprising storing the files on data storage media volumes, writing, to each of the media volumes, volume metadata that uniquely describes the respective volume, writing, with each of the files stored on the media volumes, file metadata that uniquely describes the respective file, and moving one or more of the files from one of the media volumes to another of the media volumes, wherein each of the moved files carries with it the file metadata for the respective moved file.

Brief Summary Text - BSTX (36):

In another embodiment, the present invention provides a computer-implemented system for storing files, the system comprising a processor that is programmed to control storage of the files on data storage media volumes, write, to each of the media volumes, volume metadata that uniquely describes the respective volume, write, with each of the files stored on the media volumes, file metadata that uniquely describes the respective file, and control movement of one or more of the files from one of the media volumes to another of the media volumes, wherein each of the moved files carries with it the file metadata for the respective moved file.

Detailed Description Text - DETX (3):

FIG. 1 is a functional block diagram illustrating the architecture of a hierarchical storage management system 10, in accordance with an embodiment of the present invention. System 10 will be described herein as a "directed storage management (DSM)" system inasmuch as it allows a system administrator to introduce, on a selective and reconfigurable basis, significant direction concerning storage management policies including migration. That is, the client and an administrator have the ability to direct a file to a particular location. This differs from a simple hierarchical storage management (HSM) system that handles the selection internally. The difference is that DSM system 10 provides a much broader set of data-movement policies than a typical HSM, and gives a client ways to override those policies. System 10 can be used to implement a method for hierarchical storage management in accordance with an embodiment of the present invention.

Detailed Description Text - DETX (9):

DSM server process 14 is responsible for servicing multiple concurrent requests from its various clients, i.e., user applications 28, DSM agents 22, and DSM administrator applications 24. The basic responsibilities of DSM server process 14 are (a) provide logon and file-access security for its clients; (b) handle concurrent client requests to a given file by providing simultaneous or sequential access depending on the nature of the requests; (c) translate requests for files into requests for data transfer to or from specific media/volume locations; (d) sequence data transfer requests to maximize the utilization and throughput of available devices, while providing a good quality of service to clients; (e) direct the library server process 16 to mount the required volumes at the appropriate points in time; (f) direct the volume server process 18 to establish DSM data mover process 20 for mounted volumes, and connect those processes to client data mover process 21, or other DSM data mover processes, as required; and (g) issue commands to data mover process 20, 21 to effect the requested data transfer.

Detailed Description Text - DETX (34):

File scheduler 62 sequences requests for a given file when necessary. Logically, File scheduler 62 maintains a First-In-First-Out (FIFO) queue of requests for each file. A new request to read a file can be started if there are no write requests for that file ahead of it in the queue. A write request for a file can be started only if there are no other requests ahead of it in the queue.

Detailed Description Text - DETX (39):

Selecting the volume within the Store may be postponed until a drive is available to service the request. The task of selecting a volume is given to the IO Scheduler 54. The role of the IO Scheduler 54 is to select the next IO Request to be processed. IO Scheduler 54 selects requests from the IO Request Queue 46 in order to maximize the utilization and throughput of available devices, while guaranteeing some level of service to all clients. File conflicts are resolved by File Manager 58, so the IO Scheduler 54 has a great deal of freedom in reordering IO requests. When an IO request finishes using a drive, the algorithm for selecting the next request to process takes the following into account: (1) high-priority requests that can use the drive; (2) other requests that can use the volume that is in the drive; (3) requests that can use a different volume in the same library; and (4) for sequential devices, such as tapes, the file segment position on the media relative to the current media position. A request may have originated as a high-priority request, or may have had its priority elevated based on the length of time it has been queued. Other requests that can use the volume that is in the drive may include requests that do not specify a specific destination volume, and for which there is room on the volume in the drive.

Detailed Description Text - DETX (47):

Data movers 20, 21 can be instantiated in two places: (1) on client hosts by the DSM client library 12 to read and write user files and streams; and (2) on hosts that have DSM drives by the Volume Server 18 to read and write DSM volumes. A Data Mover process 20, 21 has two communication ports that are made known to the DSM Server 14,. One port is used by the IO Scheduler 54 to issue instructions for file-related operations. Common commands include: (a) create a file; (b) delete a file; (c) return information about a file (metadata); (d) Read a file or portion of a file from media and transfer it to another Data Mover; (e) prepare to accept a file or portion of a file from another Data Mover and write it to media. The other port is used to transfer data from one Data Mover 20, 21 to another data mover. Data transfer operations always involve two Data Mover process 20, 21, and the two

processes are each made aware of the communications link. The implementation may have two connections between Data Movers, one for commands and one for data.

Detailed Description Text - DETX (49):

Database Server 26 stores information vital to the operation of DSM system 10. Database Server 26 provides secure, robust, transaction-oriented storage and efficient search and retrieval mechanisms. The information maintained by the Database Server 26 may include: (a) the DSM security database containing user IDs, passwords, and privileges; (b) physical configuration, including local and remote libraries and drives, and external media storage; (c) remote DSM servers and their communication parameters; (d) media inventory, including the location of each piece of media, and the space available; (e) file metadata, including the security attributes and media locations of each file; (t) logical grouping of media into DSM Stores, and information about each Store; and (g) policy parameters used by the DSM server and DSM agents. The database can be stored in tables and the database implementation provides a relational view of the data.

Detailed Description Text - DETX (85):

In reloading a file, a reload request typically causes a file to be copied to the highest-priority Store where the size of the file does not exceed the FileBypassSize of the Store. The source of the copy is the highest-priority Store that contains a copy of the file. Specifically, a reload request can be handled in the following manner: (a) select the reload source; (b) select the Store with the lowest value of the Priority property that contains a copy of the file as determined by the File vsResidence property; (c) if the file is not found locally and there are remote Servers configured, search the remote servers for the file; (d) select the reload destination; (e) select the Store specified by the ReloadToStore property of the source Store selected in step (a) if the source Store is local, but if ReloadToStore is null, then no reload is performed; (f) if the source store is on a remote server, then select an Initial Store on the local system; (g) if the size of the file to be reloaded exceeds the FileBypassSize of this Store, then select the Store with the next lowest value in its Priority property and continue until a Store is found where the file size does not exceed FileBypassSize; and (h) if the source and destination are the same, no reload is required, otherwise schedule a copy of the file from the source Store to the Destination Store. Normal copy and migration rules apply to reloaded files, including scheduling additional copies per the vsCopyTo. Reloading a fileset is generally equivalent to reloading all the files within the required fileset. All the files will be queued for reloading according to its location if the media type is sequential access.

Detailed Description Text - DETX (100):

DSM system 10 can be further configured to take advantage of metadata associated with each media volume and each file, thereby providing features referred to as media independence and self-describing media. Metadata can be generated for each volume in the form of a file that uniquely identifies the volume with a guid and other useful information. In this case, system 10 may generate the metadata. For individual files, however, it may be desirable for the client application to generate metadata, including a guid and information concerning the particular customer, project, or transaction associated with the file.

Detailed Description Text - DETX (101):

In this manner, the client application can pass through to DSM system 10 simply the content of the image file as a blob and the content of the metadata file as a

blob. In other words, DSM system 10 need not be concerned with the content of the image/metadata file. Rather, the client application provides sufficient information to the database server, such as a SQL server, to allow DSM system 10 to locate the file and retrieve it for direct access by the client application. The client application is responsible for the format of the image file.

Detailed Description Text - DETX (102):

Thus, for a web-based client application, the image file can be converted to a browser-recognizable format prior to submission to DSM system 10. Along with media independence, system 10 may further include a feature whereby volume metadata is stored on each physical volume to track the volume across within a local server or across a network. The metadata can be useful in tracking volumes and files, verification of the identities of loaded volumes, and database recovery.

Detailed Description Text - DETX (114):

Thus, the client application supporting the web-based workflow within web browser 74 converts each acquired image to the supported format such that the user can have direct access to the images maintained by the storage management function of DSM server 14 within the web browser. No images need to be kept directly within the client database. The screen resolution and thumbnail versions can be "locked" down by DSM server 14 on the short-term media, such as a RAID, and never allowed to migrate offline to tape. At the same time, the high resolution image may be allowed to migrate according to user-defined migration policies then in effect. In this manner, internet access to the images is as quick as possible. Also, the locked-down images are not stored by the client database server, but rather by the HSM server for direct access by the user. By migrating the high resolution image, however, storage costs for RAID are dramatically reduced. In other words, an application can be configured to store images as economically and efficiently as possible using system 10, with the potential for growth of storage capacity being unlimited and scalable.

Detailed Description Text - DETX (116):

This vault can be password and login protected. All image transmissions can be done under SSL. The vault image viewing is also secured through a virtual directory service, another sequence of logins into the storage management system 10, a Microsoft SQL Server, and the Windows NT NTFS file system itself via ACL. From the vault, the consumer can proof and create his/her own media order. That media order is placed into the a "shopping basket" for the consumer, and totaled for payment and shipping. The consumer may also send those images to a third party via internet mail. Those images stored within the vault are set on an aging algorithm where after a predetermined number of days, the images are deleted from system 10.

Claims Text - CLTX (31):

writing, to each of the media volumes, volume metadata that uniquely describes the respective volume;

Claims Text - CLTX (32):

writing, with each of the files stored on the media volumes, file metadata that uniquely describes the respective file; and

Claims Text - CLTX (33):

moving one or more of the files from one of the media volumes to another of the media volumes, wherein each of the moved files carries with it the file metadata for the respective moved file.

Claims Text - CLTX (34):

24. The method of claim 23, wherein each of the volume metadata and the file metadata is based on a globally unique identifier (guid).

Claims Text - CLTX (35):

25. The method of claim 23, wherein storing each of the files includes storing a data file containing the contents of the respective file and storing a metadata file containing the file metadata.

Claims Text - CLTX (38):

in the event the original database entries become corrupted, reconstructing the database entries using the file metadata written with each of the files stored on the respective media volume.

Claims Text - CLTX (39):

27. The method of claim 26, wherein the file metadata for the files stored on the respective media volume provides sufficient information to reconstruct the database entries access to the original database entries.

Claims Text - CLTX (42):

write, to each of the media volumes, volume metadata that uniquely describes the respective volume;

Claims Text - CLTX (43):

write, with each of the files stored on the media volumes, file metadata that uniquely describes the respective file; and

Claims Text - CLTX (44):

control movement of one or more of the files from one of the media volumes to another of the media volumes, wherein each of the moved files carries with it the file metadata for the respective moved file.

Claims Text - CLTX (45):

29. The system of claim 28, wherein each of the volume metadata and the file metadata is based on a globally unique identifier (guid).

Claims Text - CLTX (46):

30. The system of claim 28, wherein storing each of the files includes storing a data file containing the contents of the respective file and storing a metadata file containing the file metadata.

Claims Text - CLTX (49):

in the event the original database entries become corrupted, reconstructing the database entries using the file metadata written with each of the files stored on the respective media volume.

Claims Text - CLTX (50):

32. The system of claim 31, wherein the file metadata for the files stored on the respective media volume provides sufficient information to reconstruct the database entries access to the original database entries.